

It Takes a Socio-Technical Ecosystem

John D. McGregor
Strategic Software Engineering Research Group
Clemson University
Clemson, SC 29634
johnmc@clemson.edu

J. Yates Monteith
Strategic Software Engineering Research Group
Clemson University
Clemson, SC 29634
jymonte@clemson.edu

ABSTRACT

There are both technical and social issues regarding the design of sustainable scientific software. Scientists want continuously evolving systems that capture the most recent knowledge while developers and architects want sufficiently stable requirements to ensure correctness and efficiency. A socio-technical ecosystem provides the environment in which these issues can be traded off.

1. INTRODUCTION

Software is increasingly the tool of choice for tasks in scientific research, ranging from simulations replacing expensive, high risk or dangerous experiments to collecting, transforming and analyzing vast amounts of data. Scientific software has reached levels of complexity rivaling those of modern skyscrapers, yet the software is often constructed by people with little or no background in the design or sustainment of software products. The term “software ecosystem” [2] has been used to describe the interactions among organizations that collaborate on the development of a common platform and compete through the products built from that platform. The term hides an essential element in the development of scientific software - the people who provide scientific and software expertise. “Socio-technical ecosystems” [10] clearly express the idea that the people part of software development and sustainment is of equal importance to the software part of development and sustainment.

The lead author began creating physics simulations as an undergraduate and went on to spend some time at a national laboratory. Most recently a National Science Foundation grant allowed both authors to focus on software development in scientific research. We have examined research groups ranging from the students of an individual researcher to large multi-institutional groups as well as an international network of researchers in an effort to provide recommendations on scientific research software ecosystems [12]. In this very short position paper we will summarize several observations resulting from our investigations of ecosystems.

2. OBSERVATIONS

Recognize the role of system architecture Create a system architecture, even a simple one, can provide both structure to both the software and the teams implementing the system. On the other hand, being unaware or ignorant of system architecture can result in software that is hard to maintain, change and extend [9]. The architecture guides the identification of critical sections that should be assigned

to the most experienced people. Following the architecture also optimizes the lines of communication among the teams. Recognizing and being aware of the role of architecture is the difference between a software system that extends, scales and is flexible over time [1, 5, 4] and an software tool that remains in production past its expiration date.

Structure the community and establish governance Structure the community to be compatible with the system architecture. The means of structuring should encourage interactions between representatives of different organizations [9]. Establishing a governance system that recognizes the contributions of the software engineers as well as the scientists is key to sustainable scientific software. The ecosystem should be structured to prevent the dominance from a single large contributor.

Distinguish between stable and rapidly evolving knowledge Separate stable knowledge from the new knowledge being created through research results. The system architecture should be modular in places where it is likely to change and that is optimized in those areas where change is less likely. Using a platform architecture, the stable knowledge is captured in the platform and new knowledge is captured in the extensions that complete a product. In the NAMD molecular dynamics suite the maintainers keep core functionality stable while providing extension mechanisms through scripting engines that allow users to develop custom and rapidly evolving routines and analyses [14].

Make carpentry as important as results Make code quality as critical as research results. The science team focuses on a moving frontier. they conduct experiments, record results, publish findings, and push forward. Under the pressure of paper deadlines and funding reviews, code quality will have fallen by the wayside, reflected in a lack of unit tests, documentation, commenting or technical debt. Key to ensuring the long lived success and fitness of the software being developed, the software team needs to schedule time to pay back that debt by doing some cleanup work. There is value in engaging in software engineering practices such as regression testing, unit testing and test-driven development [3], as well as code review [13]. Making test artifacts, architecture, source code and requirements correct with respect to the final iteration is a useful investment to the longterm sustainability and success of the project.

Designate gatekeepers to maintain integrity of the code base Protect the integrity of the core platform by being judicious in the inclusion of new contributions. The Eclipse model as well as that of many projects gives special status to a few people who spend some of their effort

reviewing code submitted by others and who organize the contributions in satisfactory ways. This is necessary when a large percentage of the developers are graduate students, particularly when the students are not software engineering students [14, 12]. The meritocracy governance approach of Eclipse is carried into the fact that gatekeepers are voted into the position based on the quality of their contributions.

Maintain development roadmaps Build a roadmap to look into the future. Building a roadmap is a matter of thinking ahead. Scientists are thinking about sequences of experiments and software should likewise be planned ahead to guide the architects and provide a clear path of development moving towards the future. Several consortia are aware of the benefits of utilizing a roadmap to provide clear planning on the developmental trajectory on software, including the Apache Foundation and the Eclipse Foundation. Within the Eclipse Science Working Group (ESWG), the Chemclipse and DAWNSci projects follow the Eclipse Development Process (EDP) roadmap as an incubating project in order to be integrated with the Eclipse Technology Project. Similar to the EDP, the Apache Software Foundation provides a process roadmap known as The Apache Way. These processes provides a roadmap from the inception and proposal of a new project through maturity to integration and archiving in their respective ecosystems.

Develop a business strategy and plan Develop a strategy for sustainment as the research enterprise grows. A business plan should describe sources of revenue such as licensing fees and training courses. A strategy for recruiting additional collaborators should be compatible with the governance system. Others have suggested alternative models of funding from public sources, wherein software development, rather than research endeavors, is funded for maintenance, refactoring and new features [7, 8].

Be transparent Consider the structure of the organization. A productive research software community is very likely to be distributed across institutions and geography. Establishing GitHub repositories for documents, the architecture, and test cases is just as important as posting the software. Transparency and openness among the teams and organizations contributing and using the software platform is necessary regardless of whether the software is licensed as open source or not [11, 6].

3. CONCLUSION

Over the last two years we have explored a cross section of scientific software development organizations. We have identified a number of actions that contribute to successful development. In this paper we have described some of these actions and have given a very brief explanation as to why they are effective at improving development and sustainment practices. Our study has solidified our view that an explicit ecosystem strategy contributes to the success of the software development project.

4. ACKNOWLEDGMENTS

The work of the authors was funded by the National Science Foundation grant #ACI-1343033.

5. REFERENCES

- [1] S. d. S. Amorim, E. S. De Almeida, and J. D. McGregor. Extensibility in ecosystem architectures: An initial study. In *Proceedings of the 2013 International Workshop on Ecosystem Architectures*, WEA 2013, pages 11–15, New York, NY, USA, 2013. ACM.
- [2] J. Bosch and P. Bosch-Sijtsema. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83:67–76, 2010.
- [3] T. Clune, M. Rilee, and D. Rouson. Testing as an essential process for developing and maintaining scientific software. In *The 2nd Workshop on Sustainable Software for Science: Practices and Experiences*, 2014 Nov. 16, New Orleans, LA, 2014.
- [4] S. da Silva Amorim, J. D. McGregor, E. S. de Almeida, and C. von Flach G. Chavez. Flexibility in ecosystem architectures. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 14:1–14:6, New York, NY, USA, 2014. ACM.
- [5] S. Da Silva Amorim, E. Santana de Almeida, and J. McGregor. Scalability of ecosystem architectures. In *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*, pages 49–52, April 2014.
- [6] S. Darden. Transparency in software development. Technical report, Devizan, October 2013.
- [7] M. R. de Souza, R. Haines, and C. Jay. Defining sustainability through developers’ eyes: Recommendations from an interview study. Technical report, Technical Report 1111925, figshare, 2014. <http://dx.doi.org/10.6084/m9.figshare.1111925>, 2014.
- [8] R. R. Downs, W. C. Lenhardt, E. Robinson, E. Davids, and N. Weber. Community recommendations for sustainable scientific software. In *The 2nd Workshop on Sustainable Software for Science: Practices and Experiences*, 2014 Nov. 16, New Orleans, LA, 2014.
- [9] C. R. Ferenbaugh. Experiments in sustainable software practices for future architectures. In *The 1st Workshop on Sustainable Software for Science: Practices and Experiences*, 2013 Nov. 17, Denver, CO, 2013.
- [10] F. W. Geels. From sectoral systems of innovation to socio-technical systems: Insights about dynamics and change from sociology and institutional theory. *Research Policy*, 33(6-7):897 – 920, 2004.
- [11] S. Jansen, S. Brinkkemper, J. Souer, and L. Luinenburg. Shades of gray: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, 85(7):1495 – 1510, 2012. Software Ecosystems.
- [12] J. Y. Monteith, J. D. McGregor, and J. E. Ingram. Scientific research software ecosystems. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 9:1–9:6, New York, NY, USA, 2014. ACM.
- [13] M. Petre and G. Wilson. Code review for and by scientists. In *The 2nd Workshop on Sustainable Software for Science: Practices and Experiences*, 2014 Nov. 16, New Orleans, LA, 2014.
- [14] K. Schulten. Interview with Klaus Schulten. Telephone Interview, October 2013.